8Д

УДК 004.272.43

А.И. Дордопуло[1]***, И.И. Левин***[2]***, В.А. Гудков***[2]***, А.А. Гуленок***[2]
[1]Южный научный центр РАН, г. Ростов-на-Дону, Россия
[2]НИИ многопроцессорных вычислительных систем им. академика А.В. Каляева
Южного федерального университета, г. Таганрог, Россия
scorpio@ mvs.tsure.ru, levin@ mvs.tsure.ru, gudkov@mvs.tsure.ru, andrei_gulenok@mail.ru

Комплекс средств разработки прикладных программ на реконфигурируемых вычислительных системах

В статье рассматривается комплекс системного программного обеспечения средств разработки прикладных программ для реконфигурируемых вычислительных систем, включающий трансляторы языков программирования, среды разработки и синтеза прикладных схемотехнических решений, средства удаленного доступа к вычислительным ресурсам базовых модулей и ряд служебных программ. Комплекс средств разработки прикладных программ позволяет упростить процесс разработки прикладных программ для реконфигурируемых вычислительных систем на основе программируемых логических интегральных схем и сократить сроки разработки прикладного программного обеспечения.

Применение реконфигурируемых вычислительных систем (РВС) для решения сильносвязанных задач позволяет существенно повысить производительность прикладных задач различных классов по сравнению с использованием традиционных многопроцессорных вычислительных систем кластерной архитектуры.

Разрабатываемые и развиваемые более 20 лет в Научно-исследовательском институте многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета РВС представляют собой высокопроизводительную вычислительную систему, работающую по принципу конвейерной обработки потока данных [1] под управлением хост-компьютера. Отличительной особенностью таких систем является возможность изменения архитектуры системы на логическом уровне, что позволяет создавать вычислительный конвейер, соответствующий алгоритму обработки данных на уровне аппаратных устройств для решаемой задачи. Программирование РВС заключается в создании как параллельной программы, организующей потоки данных, так и наиболее эффективной конфигурации вычислительной системы, соответствующей алгоритму решаемой задачи. Поэтому РВС, с одной стороны, чрезвычайно привлекательны с точки зрения получения максимальной производительности, а с другой стороны, характеризуются очень сложным и трудоемким процессом программирования, сопоставимым по сложности с созданием новой вычислительной системы.

Высокая реальная производительность РВС, составляющая не ниже 60% от пиковой производительности системы, обеспечивается как структурно-процедурными методами организации вычислений и архитектурными принципами построения системы, так и программным комплексом средств разработки прикладных программ, обеспечивающим следующие возможности:

– программирование как структурной, так и процедурной составляющих на языке высокого уровня;
– реконфигурацию прикладных задач без участия высококвалифицированного схемотехника;
– обеспечение совместимости и переносимости проектов между РВС разных архитектур;

– масштабирование прикладной задачи при увеличении ресурса;
– удаленное использование вычислительных ресурсов РВС.

Созданный комплекс программных средств разработки [2], структура которого представлена на рис. 1, по функциональному назначению разделяется на комплекс средств разработки прикладных программ, средств администрирования вычислительных ресурсов РВС и комплекс служебных программ и драйверов.

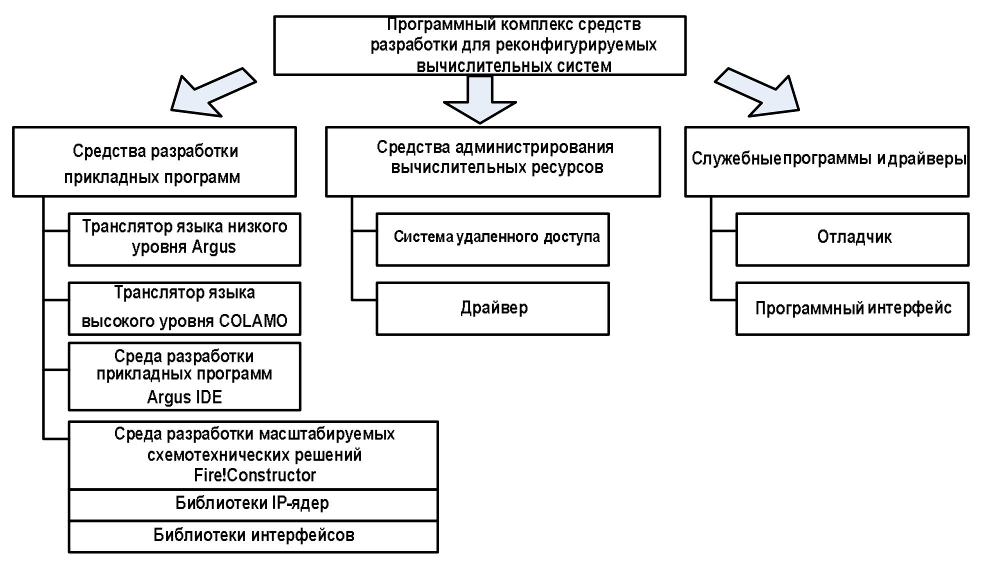


Рисунок 1 – Структура программного комплекса средств разработки для РВС

Средства разработки прикладных программ содержат: транслятор языка ассемблера Argus v. 3.0; транслятор языка программирования РВС высокого уровня COLAMO v. 2.0; интегрированную среду разработки прикладных задач Argus IDE v. 3.0, поддерживающую языки программирования Argus v. 3.0 и COLAMO v. 2.0; среду разработки вычислительных структур для синтеза масштабируемых параллельно-конвейерных процедур, оперирующую библиотекой IP-ядер и интерфейсов.

Язык программирования высокого уровня COLAMO [3-5] обеспечивает поддержку создания как структурной, так и процедурной составляющих прикладной программы, реконфигурацию прикладных задач без участия высококвалифицированного схемотехника за счет неявного описания параллелизма и переносимость прикладных задач между РВС разных архитектур за счет использования файла описания архитектуры РВС и элементов библиотеки масштабируемых IP-ядер. Транслятор COLAMO v. 2.0 осуществляет трансляцию процедурной составляющей программы, организующей потоки данных, в язык ассемблера Argus v. 3.0 и создание структурной составляющей в объектном представлении, которая автоматически передается в среду разработки масштабируемых параллельно-конвейерных процедур Fire!Constructor для синтеза конфигурационных файлов ПЛИС на языке VHDL.

Фундаментальным типом вычислительной структуры в языке COLAMO является конструкция «кадр». Кадром является программно-неделимая компонента, представляющая собой совокупность арифметико-логических команд, выполняемых на различных элементарных процессорах, обладающих распределенной памятью и соединенных между собой в соответствии с информационной структурой алгоритма таким образом, что вычисления производятся с максимально возможными параллелизмом и асинхронностью.

Кадр фактически определяет вычислительную структуру и потоки данных в РВС в данный момент времени. При этом все операции в теле кадра выполняются асинхронно с максимальным параллелизмом, а последовательность смены кадров однозначно определяется программистом.

В языке отсутствуют явные формы описания параллелизма. Распараллеливание достигается с помощью объявления типов доступа к переменным и индексации элементов массивов. Для исключения конфликтов одновременного чтения и записи ячеек памяти в пределах текущего кадра используется широко распространенное в языках потока данных правило единственной подстановки: переменная, хранящаяся в памяти, может получить значение в кадре только один раз.

Для обращения к данным используются два основных метода доступа: параллельный доступ (задаваемый типом Vector) и последовательный доступ (задаваемый типом Stream). На рис. 2 представлены программы, являющиеся граничными примерами извлечения параллелизма, и графы синтезируемых вычислительных структур.

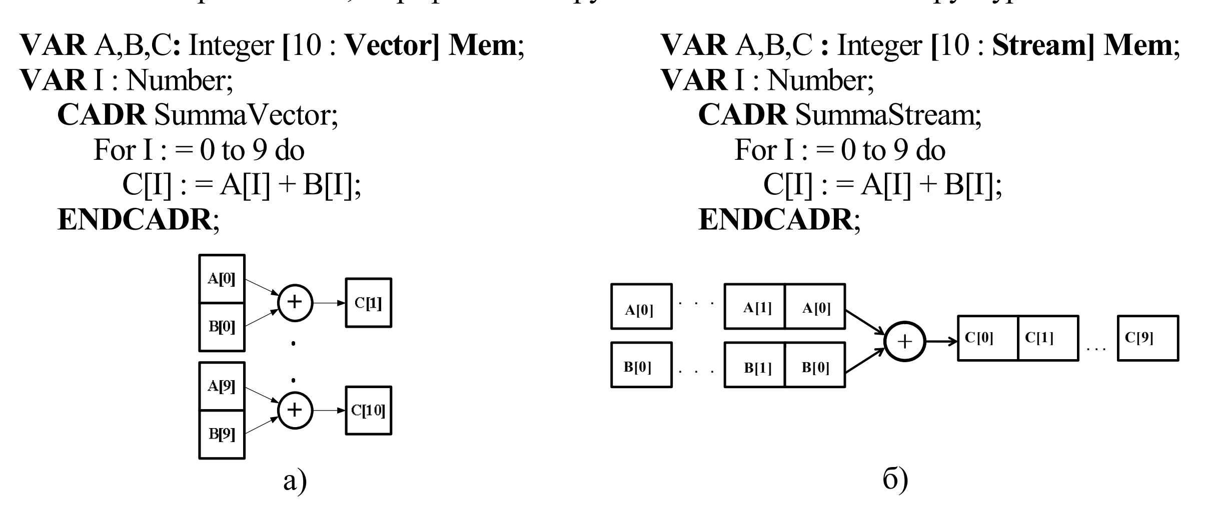


Рисунок 2 – Параллельное и последовательное сложение массивов

Тип доступа Stream указывает на последовательную обработку элементов одномерного массива, а тип Vector позволяет обрабатывать элементы одномерного массива одновременно.

Многомерные массивы состоят из множества измерений, каждое из которых может иметь последовательный или параллельный тип доступа, задаваемый ключевым словом Stream или Vector соответственно.

Применение неявного описания параллелизма за счет задания типа доступа позволяет достаточно просто управлять степенью распараллеливания программы на уровне описания структур данных и дает возможность программисту максимально просто описывать различные виды параллелизма в достаточно сжатом виде.

Трансляция программы на языке высокого уровня COLAMO состоит в создании схемотехнической конфигурации вычислительной системы (структурной составляющей) и параллельной программы, управляющей потоками данных (потоковой и процедурной составляющих).

Операторы и функции языка (сумматоры, умножители, функции сравнения, тригонометрические функции и др.), используемые в тексте параллельной программы, имеют готовые схемотехнические решения. Данные решения разрабатываются специалистами-схемотехниками в интегрированной среде разработки цифровых устройств ISE фирмы XILINX или с ней совместимых и включаются в библиотеку транслятора языка COLAMO и библиотеку стандартных примитивов среды Fire!Constructor.

В процессе работы транслятора языка COLAMO формируется информационный граф прикладной задачи из текста параллельной программы, где операторы и функции языка по определённым правилам заменяются соответствующими блоками или группами блоков из библиотеки стандартных примитивов.

Синтезированный вычислительный граф задачи передается в среду разработки вычислительных структур Fire!Constructor для укладки на множество ПЛИС РВС и обеспечения синхронизации между ПЛИС [4]. Одной из задач среды является формирование разбиения информационного графа прикладной задачи на непересекающиеся подграфы, каждый из которых будет структурно реализован в кристаллах ПЛИС выбранной РВС.

Составным элементом РВС является базовый модуль (БМ) – набор ПЛИС (DD1, DD2, …, DDN), соединённых ортогональной системой коммутаций (рис. 3). В РВС базовые модули ($БМ_0$, $БМ_1$, …, $БМ_M$) связаны последовательно, где каждый БМ (кроме первого и последнего) связан с одним последующим и одним предыдущим базовыми модулями.

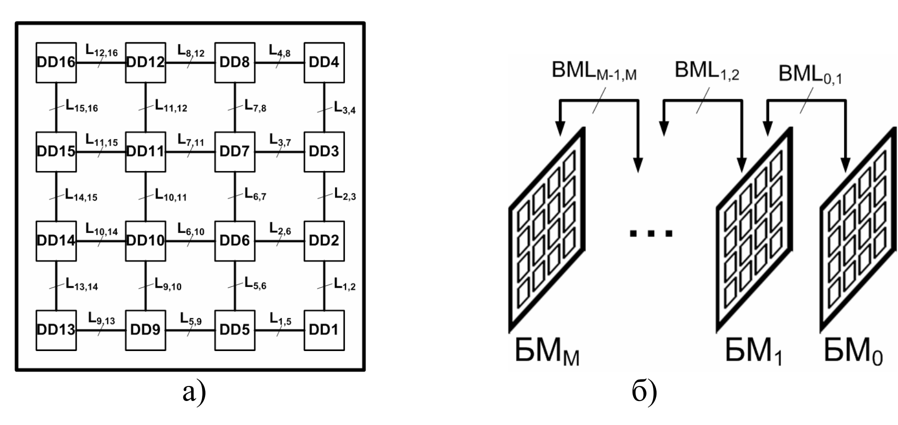


Рисунок 3 – Система коммутаций в РВС

Процесс синтеза результата разбиения информационных графов прикладных задач состоит из следующих этапов:

– решения задачи разбиения (компоновки) узлов информационного графа прикладной задачи на непересекающиеся подграфы, каждый из которых будет размещён в соответствующем БМ;

– решения задачи размещения и трассировки для узлов информационного графа в каждом БМ в отдельности и задачи трассировки связей между БМ;

– синтеза файлов VHDL-описаний и файлов временных и топологических ограничений для каждой ПЛИС, каждого БМ выбранной РВС.

Задача разбиения решается с помощью алгоритма последовательного дихотомического разрезания графа [6], с использованием разработанных эвристических методов преодоления локальных оптимумов.

Задача размещения и трассировки решается с помощью алгоритма одновременного размещения и трассировки, в котором вершины информационного графа и кристаллы ПЛИС, в которые выбранные вершины размещаются, выбираются парами последовательно. Трассировка связей производится на каждой итерации алгоритма после выбора пары размещаемой вершины и кристалла ПЛИС. Алгоритм построен таким образом, что приоритетнее считается размещение в кристалл ПЛИС, в который была помещена вершина на предыдущей итерации. Таким образом, размещение происходит как бы группами – последовательностями вершин. При этом запоминаются состояния размещения, названные «хорошими», соответствующие локальным минимумам суммы внешних связей всех вершин, последовательно размещённых в один и тот же кристалл ПЛИС. Также к «хорошим» относятся состояния, в которых в результате последнего размещения вершина была помещена в кристалл ПЛИС, отличный от предпоследнего размещения.

В случае, если не удалось выбрать пару вершина/кристалл или не удалось проложить трассы, осуществляется процедура возврата к предыдущему «хорошему» состоянию.

В процессе работы алгоритма размещения и трассировки используется многоуровневая схема [7], идея которой заключается в последовательном прохождении 3 этапов (рис. 4): укрупнение узлов исходного графа, размещение укрупнённого графа и восстановление размещённого графа до уровня исходного графа.

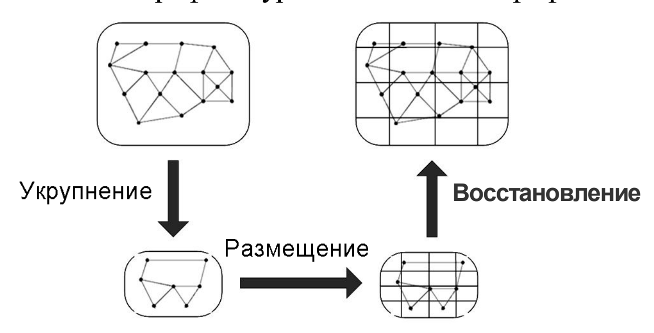


Рисунок 4 – Многоуровневая схема размещения графа

На этапе укрупнения находятся непересекающиеся группы узлов графа, которые выгодно размещать в одном и том же кристалле ПЛИС. При этом на группы могут накладываться ограничения по размерности группы и по количеству внешних связей группы. Далее из каждой группы исходного графа формируется по одному мультиузлу укрупнённого графа. Классическим алгоритмом укрупнения исходного графа является алгоритм попарного стягивания, на каждой итерации которого по определённым критериям происходит выборка пар смежных узлов, которые будут объединены в мультиузлы, и полученный граф поступит на следующую итерацию алгоритма. Таким образом, на каждой итерации алгоритма количество узлов исходного графа уменьшается вдвое. Алгоритм останавливается, когда будет достигнуто заданное количество узлов в укрупнённом графе.

Для этапа укрупнения исходного графа был разработан модифицированный алгоритм попарного стягивания вершин графа, на каждом этапе которого находится единственная пара узлов графа, которая объединяется в мультиузел. Главным критерием оценки качества сформированного мультиузла является отношение количества внешних связей к внутренним. Алгоритм учитывает ранее размещённые узлы графа в ПЛИС и позволяет формировать вокруг размещённых узлов мультиузлы с заведомо известными их размещениями.

На этапе размещения многоуровневой схемы среда работает согласно приведённому выше алгоритму одновременного размещения и трассировки.

На этапе восстановления размещённый укрупнённый граф разворачивается до уровня исходного графа. При этом узлы исходного графа, вошедшие в одну группу, будут размещены в том же кристалле ПЛИС, в котором был размещён мультиузел укрупнённого графа.

Третьим этапом синтеза результата разбиения информационных графов прикладных задач является *синтез файлов VHDL-описаний и файлов временных и топологических ограничений для каждой ПЛИС, каждого БМ выбранной РВС*. Данные файлы подключаются к проекту в интегрированной среде разработки цифровых устройств ISE фирмы XILINX, в котором содержатся все задействованные в формировании информационного графа задачи, схемотехнические решения вычислительных и интерфейсных блоков, и для каждой микросхемы создаются конфигурационные файлы ПЛИС.

Среда Fire!Constructor упрощает создание масштабируемых структурных решений и сокращает время разработки за счет автоматизированного выполнения следующих трудоемких процедур:

– согласования входов и выходов совместно работающих ПЛИС (ucf-файлов);

– автоматической синхронизации информационных потоков при размещении функциональных устройств в едином вычислительном контуре, расположенном в различных кристаллах ПЛИС;

– автоматического обеспечения сбалансированного размещения функциональных устройств по различным ПЛИС.

Общая взаимосвязь транслятора языка COLAMO, среды Fire!Constructor и синтезатора конфигурации ПЛИС в рамках комплекса системного программного обеспечения при создании многокристального схемотехнического решения для базового модуля РВС представлена на рис. 5.

Такой подход к программированию реконфигурируемых вычислительных систем позволяет освободить программиста от построения графа задачи в виде функциональных библиотек в среде Fire!Constructor и синхронизации потоков данных в РВС, сократив время создания параллельных программ для РВС в 3 – 10 раз, и исключить участие специалиста-схемотехника при разработке параллельных прикладных программ.

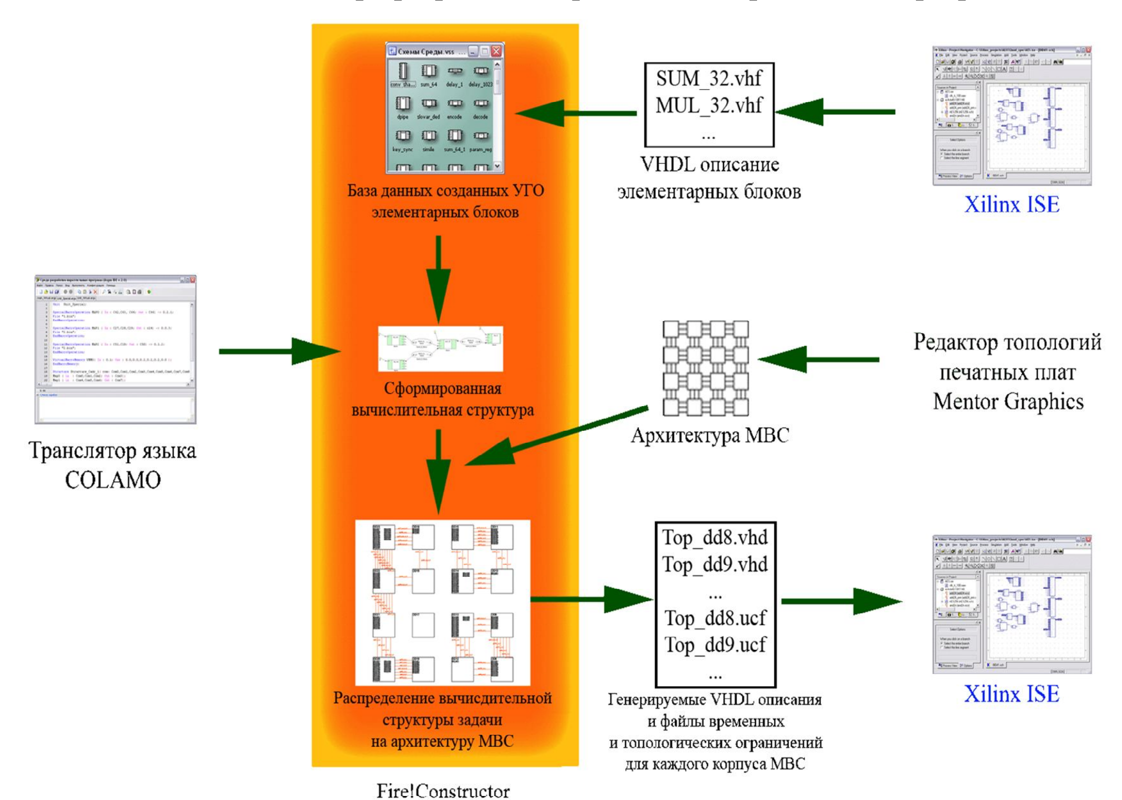


Рисунок 5 – Взаимосвязь компонент комплекса при создании многокристального схемотехнического решения для базового модуля РВС

Язык структурно-процедурного программирования Argus представляет собой низкоуровневый язык (ассемблер), предназначенный для описания процедурной составляющей прикладной параллельной программы РВС [2], [3]. Программа на языке Argus организует потоки данных на уровне команд контроллеров распределенной памяти, обеспечивая их синхронизацию. Программирование на языке Argus, как и на любом языке ассемблера, требует от программиста обширных знаний в области аппаратного

обеспечения РВС, команд контроллеров распределенной памяти и взаимосвязей между структурными элементами системы.

Интегрированная среда разработки Argus IDE предназначена для интерактивной разработки параллельных программ на языках высокого уровня COLAMO и языке ассемблера Argus в едином языковом пространстве. Среда Argus IDE, объединяя в своем составе трансляторы языков COLAMO и Argus, обеспечивает эффективную разработку масштабируемых параллельных программ для РВС.

Созданное параллельное решение прикладной задачи в виде загрузочного модуля РВС с помощью драйвера загружается в базовый модуль РВС. Драйвер базовых модулей обеспечивает программную поддержку функций непосредственного доступа к высокоскоростному аппаратному интерфейсу, поддерживающему пакетные режимы работы и обеспечивающему механизмы прямого доступа к физической памяти управляющего компьютера. Кроме того, драйвер выполняет поддержку режимов энергосбережения.

Комплексная отладка программ, функционирующих на множестве базовых модулей РВС, осуществляется с помощью отладчика параллельных программ, обладающего следующими возможностями:
– поддержкой точек останова для контроллеров памяти, в том числе и условных;
– обеспечением полного доступа к памяти базовых модулей в любой момент времени;
– обеспечением выполнения и возможности настройки команд пуска, останова, выгрузки промежуточных результатов.

Для удаленного доступа и управления вычислительными ресурсами РВС разработана система удаленного доступа, которая состоит из сервера, обрабатывающего удаленные заявки на использование вычислительных ресурсов и поддерживающего очередь заявок, и клиента, формирующего заявки на основе команд пользователя. К функциям системы удаленного доступа относятся функции включения, выключения, остановки и запуска как отдельных базовых модулей, так и блоков, стоек и РВС в целом (рис. 6).

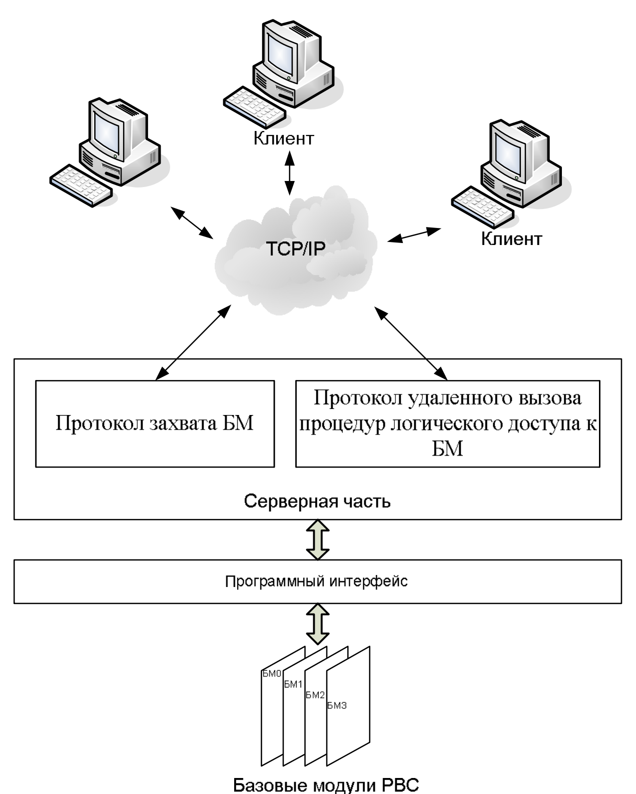


Рисунок 6 – Структурная схема системы удаленного доступа

Для доступа к вычислительным ресурсам РВС и основным функциям разработанных программных средств из других систем разработки создан программный интерфейс доступа к базовым модулям РВС. Все системное и прикладное программное обеспечение основывается на использовании программного интерфейса, что позволяет обеспечить переносимость и масштабируемость (расширяемость) системного и прикладного программного обеспечения независимо от низкоуровневой программно-аппаратной реализации функций доступа к базовым модулям.

Созданный комплекс системного программного обеспечения позволяет создавать эффективные прикладные программы для РВС при решении задач различных предметных областей без знания специфики организации структурно-процедурных вычислений.

Созданные средства разработки прикладных программ для реконфигурируемых вычислительных систем обеспечивают удобство программирования сложных практических задач на РВС и сокращают время разработки прикладного решения в 3 – 5 раз, обеспечивая при этом автоматизированный перенос структурного решения с одной архитектуры РВС на другую.

Литература

1. Воеводин В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – С.-Пб. : БХВ-Петербург, 2002. – 599 с.
2. Дордопуло А.И. Семейство многопроцессорных вычислительных систем с динамически перестраиваемой архитектурой / А.И. Дордопуло, И.А. Каляев, И.И. Левин, Е.А. Семерников // Материалы Четвертой Международной научной молодежной школы «Высокопроизводительные вычислительные системы». – Таганрог : Изд-во ТТИ ЮФУ, 2007. – С. 68-74.
3. Каляев А.В. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений / А.В. Каляев, И.И. Левин. – М. : Янус-К, 2003. – 380 с.
4. Каляев И.А. Реконфигурируемые мультиконвейерные вычислительные структуры / [И.А. Каляев, И.И. Левин, Е.А. Семерников, В.И. Шмойлов] ; под общ. ред. И.А. Каляева. – Ростов/Д : Изд-во ЮНЦ РАН, 2008. – 320 с.
5. Левин И.И. Язык параллельного программирования высокого уровня для структурно-процедурной организации вычислений / И.И. Левин // Труды Всероссийской научной конференции. – М. : Изд-во МГУ, 2000. – С. 108-112.
6. Мелихов А.Н. Применение графов для проектирования дискретных устройств / А.Н. Мелихов, Л.С. Берштейн, В.М. Курейчик. – М. : Наука, 1974. – 304 с.
7. Kumar Karypis. Multilevel algorithms for multi-constraint graph partitioning. Technical Report TR 98-019, Department of Computer Science / Karypis Kumar, V. Kumar. – University of Minnesota. – 1998.

А.І. Дордопуло, І.І. Левін, В.А. Гудков, А.А. Гуленок

Комплекс засобів розробки прикладних програм на реконфігурованих обчислювальних системах

У статті розглядається комплекс системного програмного забезпечення засобів розробки прикладних програм для реконфігурованих обчислювальних систем, що включає транслятори мов програмування, середовища розробки і синтезу прикладних схемотехнічних рішень, засоби віддаленого доступу до обчислювальних ресурсів базових модулів і ряд службових програм. Комплекс засобів розробки прикладних програм дозволяє спростити процес розробки прикладних програм для реконфігурованих обчислювальних систем на основі програмованих логічних інтегральних схем і скоротити терміни розробки прикладного програмного забезпечення.

A.I. Dordopulo, I.I. Levin, V.A. Gudkov, A.A. Gulenok

System software complex of applied programs development tools for reconfigurable computer systems is discussed in the paper. System software complex consists of programming language translators, environment of applied circuit solutions development and synthesis, tools for remote access to basic modules computational resources and a number of utility programs. Complex of applied programs development tools allows to simplify development of applied programs for reconfigurable computer systems on the base of FPGAs and decrease time of applied software development.

Статья поступила в редакцию 22.06.2009.